Move Semantics Workshop Solutions

Copyright © James Raynard 2021

Constructor

- We know that std::thread is move-only, so this will have to be a move constructor

```
thread_guard(std::thread&& t) : t(std::move(t)) {}
```

- We will make the constructor explicit, to avoid conversions
- This is unlikely to be a problem here, but is good practice

```
explicit thread_guard(std::thread&& t) : t(std::move(t)) {}
```

- Since we have defined a constructor, the compiler will not generate a default constructor for us

```
explicit thread_guard() = default;                    // Not required, but may be useful
```

Copyright © James Raynard 2021

Copy constructors

- The class must not be copyable, so we prevent the compiler from generating copy operators

```
thread_guard(const thread_guard&) = delete;
thread_guard& operator=(const thread_guard&) = delete;
```

- The class must be moveable
- Since we have defined a destructor, the compiler will not generate move operators, so we must add them

```
thread_guard(thread_guard&&) noexcept = default;
thread_guard& operator=(thread_guard&&) noexcept = default;
```

Copyright © James Raynard 2021

Camera class

- We need to deduce the interface of the Camera class
  - We know that it has public member functions take_picture() and replace()
  - We know that it has a constructor that takes a pointer to MemoryCard, and Camera has custody of this pointer
  - Camera needs to implement the "rule of five" member functions since it is managing a resource
  - Camera is moveable but not copyable. This allows it to maintain custody of the pointer
  - To make it uncopyable, the copy operators must be deleted

Copyright © James Raynard 2021

Camera class definition

```
class Camera {
   private:
      MemoryCard *m;
   public:
      Camera(MemoryCard *m) : m(m) {}
      Camera(const Camera&) = delete;
      Camera(Camera&& other) noexcept;
      Camera& operator=(const Camera&) = delete;
      Camera& operator=(Camera&& other) noexcept;
      ~Camera();
      void take_picture();
      void replace(MemoryCard *);
};
```

Copyright © James Raynard 2021